Issues in the Management of Heterogenous Computer Networks Operated in Multiple, Variable, Layered Domains.

N.W.Dawes, B.Pagurek

July 15, 1994

SCE-94-19

Department of Systems and Computer Engineering
Carleton University
Ottawa, Ontario, K1S5B6

Supply and Services Canada ref:

W2207-02-AF03/005SV

copyright:
Her Majesty the Queen in Right of Canada (1994)
as represented by the Minister of National Defence

ABSTRACT

Heterogeneous computer networks managed by several operational authorities are abnormally fragile. The fragility is due to the gaps in the knowledge of the operations managers. The central issue is therefore the management of these gaps; in other words, managing the uncertainty. In detail, failures in such networks are more complex than usual and more frequent than they might be for several reasons. The lack of standards to permit the ready transfer of monitoring and diagnosis information between the different managers is one reason. The frequent current impossibility of direct transfer of the information, even if standardised, is a second. The difficulty of altering the management systems to support new devices and new services is a third. The information gathered by several sources frequently needs to be both coordinated and viewed from several perspectives and such tools are not available. Finally the proactive analysis needed to anticipate and reduce the frequency of failures is not,well supported by tools and is not perceived as cost effective in many network operations centres. This report describes work being done to address these issues, some general solutions and some particular solutions. Standards work, techniques for exploiting standards and architectures to support such techniques are also described.

The following report describes research carried out by Carleton University on behalf of the Department of National Defence. This work was funded under contract W2207-02-AF03/005SV and was carried out in 1993 and 1994. Thw work was performed by Dr. N.W. Dawes and Dr. B. Pagurek with Mr. V. Taylor as the DND Technical Advisor.

TABLE OF CONTENTS

GLOSSARY of ACRONYMS.....3

A. INTRODUCTION.....4

B. THE SOURCES OF PROBLEMS.....4

1: Network systems problems.....4

1.1: Network systems problem variety.

1.2: Network views.

2: User problems.....6

3: Operator problems.....7

3.1: Planning to solve problems.

3.2: Solving problems.

C. STANDARDISATION.....8

1: Standard representations of information for network management

2: Conclusions.

D. MANAGEMENT WITHIN STANDARDS.....10

1: Management Architectures for Heterogeneous Networks.....10

1.1: Manager/Agent Architectures

1.1.1 The Blackboard Architecture

1.2: Agent Usage

1.3: Manager/agent usage

1.4: Implementation Requirements

1.5: Achieving Extensibility

1.5.1: Community String Multiplexing

1.5.2: The SMUX Protocol

1.5.3: The DPI protocol

1.5.4: API and Programmatic Agent Extension Methods

2: The Pegasus Blackboard.....19

2.1: Blackboard control

2.2: The blackboard database.

2.3: Blackboard tests

2.4: KS Organization.

2.5: Summary

2.6: Evolution to a Blackboard Implemented Manager/agent

3: Service Management Architecture.....25

3.1: The Internet Approach

3.1.1: The Service Management Architecture

3.1.2: The SNMP Service MIB

3.2: An OSI Approach

3.2.1 Management Services

3.3: The ODP Approach

3.2.2 Summary

E. CONCLUSIONS.....31

1: The Origins Of Issues

2: Where Knowledge is Usually Weak

3: How to Gather The Knowledge

4: How to Use The Knowledge

F. REFERENCES.....34

G. APPENDIX 1 Examples of Practical Problems.....34

H. APPENDIX 2 Complexity.....39

GLOSSARY of ACRONYMS

AI	Artificial Intelligence
API	Application Program Interface
ARP	Address Resolution Protocol
ASN.1	Abstract Syntax Notation One
ATM	Asynchronous Transfer Mode
BANES	Basic Analysis in Nested Evidence Space
CMIP	Common Management Information System
CNM	Customer Network Management
CPN	Customer Premises Network
CPU	Central Processing Unit
CRTC	Canadian Radio Television Commission
DCE	Distributed Computing Environment
DMI	Desktop Management Interface
DPI	Distributed Protocol Interface
FDDI	Fibre Distributed Data Interface
FTP	File Transfer Protocol
IAB	Internet Architecture Board
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPC	Interprocess Communication
KS	Knowledge System
LAN	Local Area Network
MAN	Metropolitan Area Network
MIB	Management Information Base
MIF	Management Information Format
MTBF	Mean Time Between Failure
MUX	Multiplexor
M2M MIB	Manager-to-manager MIB
NOC	Network Operations Centre
NMS	Network Management Station
ODP	Open Distributed Processing
OS	Operations Support
OSI	Open Systems Interconnection
PDU	Protocol Data Unit
PN	Public Network
QoS	Quality of Service
RMON MIB	Remote Monitoring MIB
RPC	Remote Procedure Call
SMDS	Switched Multimegabit Data Service
SMUX	SNMP Multiplexing Protocol
SNMP	Simple Network Management Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VPN	Virtual Private Network
WAN	Wide Area Network

A. INTRODUCTION

Heterogenous computer networks have three major user sets, the end users, the network operators and the network managers. The goals of each set are not compatible and tensions between these must be arbitrated by the managers. The users want immediate access for the lowest cost, zero errors and trivial simplicity of use. Operators want no faults and no problems. Managers want to optimize the cost effectiveness of the network operation, striving to maximize return on investment. We can see the tension between the view by example. Managers wish to continue using equipment until its support cost with respect to its profit becomes unacceptable. Users of such equipment will perceive growing problems with its use and will become increasingly dissatisfied. Operators will grow accustomed to the equipment problems and find its increasing level of problems manageable as the problems become more and more familiar. Operators will also get aggravated by the increasing number of problem calls by users. Network operations centres invariably display aspects of these tensions.

The issues of network management therefore fall into several groups. In this report we have divided them into network systems problems, user problems and operator problems. The issue of standardisation emerges from these and is discussed next. Management and standards are then considered. The appendices contained worked examples of specific problems and complexity.

B. THE SOURCES OF PROBLEMS

1: Network systems problems.

At least two major issues have emerged with respect to network systems problems: the staggering variety of problems possible and the requirement of managers to view networks in different ways.

1.1: Network systems problem variety.

A complete list of network systems problems cannot be compiled at any time for large, useful, heterogenous networks. This is a major issue. The number of lines of code in subsystems (eg: network routers) varies between about 100,000 to about 1,000,000 lines. The number of differently configured or constructed subsystems in such networks is often in the thousands. The number of different states/interactions in protocol layers typically runs into the hundreds. The combination of the these factors with variability introduced by other factors (eg: load, time of flight) means the number of ways a single fault in a single line of code can manifest itself are frequently incomputably large. Finally, the dynamic nature of networks and the rapid proliferation of new devices

further increases such numbers. This means that network problems can never be completely itemized.

We are aware of no attempts being made to completely list all possible faults in complex heterogenous networks. Previous homogeneous networks with long-lived systems (eg: DATAPAC) have had very full lists of possible faults assembled.

The almost certain absence of such complete lists means that only categories of faults can be produced. This is common practise nowadays, although not often considered in these terms. For example, system developers routinely include traps and system alarm detection code. Such traps report, for example, 'queue overflows' or 'retry limit failures'. These are in fact generic symptoms of a wide variety of underlying faults. The faults may be in the code (implementation error) or in the algorithm (design error) or in the configuration (installation error). In other words, each system will report on a set of problems, those that the developers both anticipated and thought worthy of reporting. In practise, these reports often define very easily found faults. Unforeseen problems are often those that require the highest skill to diagnosis and have the most severe, sudden impacts. Furthermore, since systems are built with significant redundancy, symptoms related to a problem are often masked or concealed, sometimes for months. This seriously complicates the discovery of the causes of problems

This issue can be summarised as follows. Systems in heterogenous computer networks can fail in unpredicted ways. Such failures can often be masked for long periods of time, showing vague symptoms. The variety of such failures is inconceivably large. One main implication follows from this. The more stable a computer network is, the more reliable and predictable it is likely to become. Network managers should therefore explicitly trade off the cost reduction of adding new, cheaper, faster systems against the certain increase in technical problems due to the increase in overall uncertainty in the network system considered as a whole.

A generic list of network system problems is often constructed as follows. This list gives a generic type of fault and then one way in which it can be caused. In other words, it correlates user impacting symptoms to possible subsystem failures.

Connection making faults (eg: caused by name server file inconsistencies).

Loss of an active connection (eg: caused by Router overload by ARP storm).

Abnormally low data rate (eg: caused by misconfigured FTP).

High error rate (eg: caused by abnormally large packet size in

FTP).

1.2: Network views.

Managers and operators of heterogenous networks with multiple, overlapping,layered, dynamic domains must have access to many different views of the network. They need this variety of views in order to focus attention on one particular aspect or dimension of problems, mapping other (hopefully irrelevant dimensions) onto invisible surfaces. For example, a manager may want to view a network by unused traffic capacity, or by topology at an OSI level. The number of views that are useful to a particular network are probably a subset of generically useful views. We expect that the total number of useful views will probably not be excessive. However, provision must be made to allow new views to be constructed from existing data sets at any time. This means that the data about the network must be accessible as if it were a database. This has significant implications for data compatibility, integrity, validity, completeness, time stamping and history collection.

2: User problems.

User problems are a combination of human errors and equipment errors. Reducing or controlling the frequency of user problems is usually a prime goal of the network management group. (The other goals are usually network efficiency measures and network management cost effectiveness measures). User problems however include not only faults related to the network but also faults due to inadequate training of users and incompetent or abusive use of the network by users. Unfortunately user problems are sometimes also produced by network management errors. Network management usually involves proactively reducing the incidence of incompetent or abusive use by providing training.

User problems are usually inadequately reported, because the user does not know enough about the network to completely describe the problem. The operator taking the problem call is often not expert enough to analyze the information being given about the problem to ask sufficient questions directly to complete the problem description while the user is contactable. This situation, common in telephone companies, has lead to them producing more and more automated remotely operable diagnostic facilities. The ultimate goal there is to be able to initiate diagnosis while the user is still in direct contact with the operator (ie: within a few seconds of the user's first call into a single contact point). This requires, in a homogenous network, planning and preprovisioning of systems. In a heterogenous network this requires that each subnetwork provide such systems, that the diagnostic systems be compatible and that a single point of contact (for the user

complaint call) can initiate diagnosis at each subnetwork level (or domain) practically simultaneously. This requires more than planning and provisioning, it implies standardisation and cooperation amongst the domain management systems. It also raises serious issues of security control, liaison, compatibility between the network operations centres and network monitoring demarcation boundaries for each such centre.

In some heterogenous networks cooperative exchange of domain management is incompatible with the security requirements of the network users. Regulatory constraints may also prevent such exchange from occurring freely. This problem can be compared to the following mathematical analogy. Let there be three variables in three equations 1.1 to 1.3 below.

```
a + b + c = 6        ..... 1.1
c = 3                ..... 1.2
2a + b    = -4       ..... 1.3
```

However, each equation is known to a different domain manager and the manager of 1.2 may not let 1.1 and 1.3 know that c = 3. A satisfactory solution to this issue clearly cannot let 1.2 solve the problem and tell 1.1 and 1.3 the values of a and b. This analogy therefore corresponds closely to the serious and very difficult issues raised by security and regulatory requirements. However, since it is now posed in a form which is quite amenable to analysis and logic, a practical resolution or amelioration may be possible.

3: Operator problems.

There are two facets to the issue of operator problems, planning to solve them and solving them.

3.1: Planning to solve problems.

It has been observed that very great majority of network problems are more easily diagnosed the closer one gets to the problem. In other words, the localisation of problems is key to reducing their difficulty. This means that diagnostic capabilities must either be placed closely spaced in the network or that remote diagnostic agents must be able to gather diagnostic information on demand. In practise the latter method is often impracticable, for example one cannot run an Ethernet analyzer (which checks every packet on am Ethernet) from a remote site, passing all the packets from the Ethernet to the remote site. Clearly filtered, preprocessed data can be so transmitted, but the raw data must be preprocessed. With the steady reduction in hardware, local monitoring and partial diagnostic facilities should be made pervasive. However, in order to allow the integration of

information from such facilities to solve problems that might span network management domain, again the diagnostic information must be standardised. However local domain managers will not perceive such standardisation as being as pressing a need as managers whose networks span domains.

3.2: Solving problems.

Operators wish to solve problems rapidly and want there to be as few problems as possible. Since the latter goal is largely dependant on others (except when failing to repair a device causes ripple problems), the aim for operators is to optimize problem solving. Some problems in big heterogenous networks now strain the upper limits of the best human brains trained over many years. Such people are not the normal run of operators. This means that problems now occur which are beyond the capabilities of on duty operations staff. In more complex, variable networks, we should expect this situation to get worse.

There is one trend which will slow the increasing frequency of unsolvable problems (for onduty staff). The steady increase in competence of standard network diagnostic tools (eg: LAN analyzers) means that the tools can provide the answers to many problems, if they are used correctly. For example, the rapid increase in data capacity of lines (eg: broadband or fibre) means that use of the current generation of passive monitor will become increasing ineffective without significant expertise on the part of the operator. Operations staff will therefore become trained to use these increasingly complex monitoring and diagnostic tools to solve problems. Naturally this will place increasing demands on operators as well as training them to higher competence. However, the increasing complexity of these monitoring and diagnostic tools will place an upper limit to their utility.

The step beyond this is to have tools which automatically diagnose faults, probably using the complex analyzers to do so. This progression to automated fault detection, reaction and diagnosis appears the only solution which offers promise of actually restricting the complexity of the operators jobs. However, in heterogenous networks, the integration of diagnostic devices (most of which require human operation at present) will certainly require standardisation of diagnostic inputs, operation and outputs.

C. STANDARDISATION

1: Standard representations of information for network management

Unless information about network devices is standardised,

heterogenous network management will become increasing complex and less effective. At present standardisation is being applied to some information, but not to others. Data about traffic rates, errors of detailed equipment in various forms is being thoroughly addressed by IAB/IETF standard's bodies - defining MIBs for almost everything. Data about hypotheses, diagnoses and data supporting the reasoning for these have no standards and no standard bodies appear to be addressing this.

Standard representations of hypotheses, diagnoses and the supporting evidence have to be applicable both to a range of devices and to past and future versions of such devices. It is unrealistic to expect to rebuild diagnostic systems just to handle each new device type added to a network. Such a diagnostic system would constantly be being changed and would therefore be hopelessly error prone. Automated diagnostic systems must be more stable than the networks they handle. The standards defining the reasoning information must therefore be more stable than those which define the devices. This implies that the standards defining reasoning information must permit extendability so that existing code can handle future versions of devices without change. This implies existing code can also handle previous versions of devices without change.

The Pegasus project[1, 11] is perhaps the first to have to have produced a probable solution for this standardisation problem. It chose a totally free form of description (three orders of indirection) which means any application can read whatever any other can write, provided the reader knows the sense of the labels. The key idea here is that only the labels referring to objects are defined. The field descriptions are defined by the writer. Should other writers wish to extend these fields, readers can read old and new forms with the same code. This generalisation has not be proven to work in operations (only in extensive simulations and trials), but no contrary examples for its use have yet been found. It is worth noting that although the three orders of indirection implementation method was chosen for Pegasus, this does not mean that simpler or more appropriate implementation methods may not exist.

Since coordination of local diagnoses will be performed in some hierarchical manner, local agreement on data fields can be managed within a structure that permits many variations in the fields to exist at the same time within the network. This is probably essential for rugged operations of the diagnostic methods. Standards on the labelling of objects should be produced. Happily, the standards for MIBs include naming most of the physical objects required. What remains is the definition of labels for hypotheses, diagnoses and reasoning.

2: Conclusions.

The problems of managing such networks are getting worse. Automated tools are helping but their use is getting more complex as well. Integration of the present generation of automated tools to provide hypotheses and integrated views to operators and managers is required. Standardisation of diagnoses, hypotheses and reasoning is required. These issues seem to loom over the future of heterogenous computer network management. When the extra complexities for managing multiple, dynamic, layered domains are factored in, satisfactory resolution of these issues will be required much earlier than in less demanding networks. Finally, very serious and difficult issues may be raised by the restriction of access to network management information due to security or regulation. Again, these will surface early in the more demanding networks.

D. MANAGEMENT WITHIN STANDARDS

1: Management Architectures for Heterogeneous Networks

Management of heterogeneous enterprise networks is becoming increasingly complicated because of:

1. The sheer complexity of networks containing devices from many manufacturers using many protocols both at the operational and management levels

2. The size of networks which are growing at an amazing rate and which for even moderate sized enterprises commonly span several continents and contain thousands of nodes.

3. The dynamic nature of modern networks in that new types of physical networks (FDDI, Fast Ethernet, ATM, etc. etc.) are being created continually and at an increasing rate

4. New services (Frame Relay, SMDS, Multimedia) are arising rapidly and in addition the orientation of management for future networks is concentrating more and more on service orientation than on the more traditional device orientation

With regard to complexity, since nothing we do can eliminate complexity, we seek methods for bringing such complexity under control by identifying the factors that contribute to such complexity so the appropriate managerial attention can be applied where it pays off the most. Moreover if complexity issues can be pinpointed at the design stage then such problems can be reduced considerably. This is addressed in Appendix 2 of this report.

With regard to size, networks of any reasonable size are impossible to manage in a completely centralized fashion from a single location. It is therefore necessary to consider some kind of

distributed management scheme where management duties are delegated to intermediate or mid-level managers. This is even more important in heterogeneous networks where the division of effort is not solely to distribute the load but also because local expertise must be employed to manage specialized subnetworks and interdomain boundaries exist between jurisdictions. Hierarchical management is most common but the often mentioned concept of the manager of managers has not really taken hold. Even today most distributed management implementations are proprietary although some effort is underway to standardize such approaches. We seek easy to implement distributed architectures that also can be adjusted to accommodate change and extension.

With regard to the dynamic nature of networks, we seek methods to allow networks and managers to accommodate such change. Aside from new hardware, new diagnostic techniques and new monitored variables must be accommodated for the network to be manageable. This is supposed to be the era of network management applications (diagnostic algorithms, performance analysis, health profiling and problem tracking etc.) in addition to the management protocols which facilitate management communication. The applications simply are not appearing in useful numbers for several reasons. First of all the management protocols themselves are too complex or are still too dynamic to provide the necessary underpinnings. Standardization helps in assuring broader applicability of developed tools but is a slow process. Perhaps the greatest inhibition to developing applications is that the large effort is not worth it because things change too fast and the applications will quickly become obsolete. To overcome these inhibitions it is necessary to develop flexible and adaptive management tools while at the same time accommodating the increasing trend to standardization. The management tools we are talking about must be flexible and allow for a variety of distributed management frameworks. They must be extensible by permitting dynamic introduction of new perhaps experimental diagnostic techniques and the introduction of new supporting network management information (monitored variables and aggregate information i.e. new MIBS). Some of these issues are addressed in the second part of this report.

With regard to the trend to service orientation we examine the need and tools for interdomain communication required to permit end user management of WAN services. While interdomain communication for the purpose of diagnosis is available to a certain extent between carriers in a TMN environment, traditionally carriers have provided minimal information to enterprise managers to facilitate local control of end-to end services. This has resulted in finger pointing when things go wrong. Enterprises have however been demanding more and more control with the result that efforts are under way to provide service management and provisioning across boundaries.

1.1: Manager/Agent Architectures

We consider here the problem of handling the dynamic nature networks. That is we consider methods for building managers and agents which are extensible. By that we mean new MIBs can be added dynamically as well as new cooperating knowledge systems for diagnostic and other management functions.

1.1.1 The Blackboard Architecture

The Carleton blackboard architecture[10,11] was originally conceived of as a vehicle for integrating diverse reasoning agents, called knowledge systems, independent of any management protocol. Its origins lie in the AI blackboards, usually written in Lisp, for advanced reasoning experiments. Our implementation in 'C' currently has many KSs including the Dragon simulator, the Banes fault locator, and other experimental expert system, neural network, and pattern recognition tools. The blackboard architecture has many advantages.

- it is particularly useful for asynchronous event driven applications and real time systems
- it provides concurrency through the independent KSs
- it is dynamically extensible in that new KSs can easily be added without disrupting the operation of the system or the other KSs.
- it is easy to add KSs as was done by many inexperienced student programmers.
- it has proved to be very robust and efficient.

More recently we have come to believe that a blackboard architecture is very well suited to the needs of extensible management agents in general and mid-level manager/agents within the SNMP or CMIP domains. For example this architecture could be advantageous for a new expanded data-collector providing more value added services as it passes information on to the higher level manager (NOC). We visualize that this would be the location for significant trend analysis, local diagnosis, reasoning, and filtering functions. As indicated earlier this includes potential RMON and other probe management.

One of the reasons that the blackboard paradigm appears to be particularly useful is because it would facilitate the implementation of agent-subagent constructions often used to increase the capability and flexibility of an agent without impacting too much on the agent itself. This is done by having the agent contract out or dispatch much of its work to subagents. Proxy agents fall into this category as well. In designing such a combination, decisions have to made as to:

- The granularity of the dispatching to be used, per PDU or per

varbind and the method of multiplexing. For example community strings and the new SNMPv2[3] contexts appear in the SNMP wrapper around the PDU and can be used to dispatch the PDU without looking inside. ON the other hand the agent can parse the PDU and dispatch requests individually for each variable in the binding list.

- The means for accommodating the new features of SNMPv2 which have added a great deal of complexity. In particular the manager-to-manager MIB must be supported to facilitate distributed management and its deployment in an extensible way is very complicated.

- The method of communication between agent and subagent with pipes, sockets, APIs using dynamic linking, RPC, and shared memory as possibilities.

- The method of configuration of responsibility can be done by dynamic registration or by static configuration files. This can impact memory usage which is a major consideration.

What is missing today in most network management systems is a true diagnostic capability. It is still necessary, for most difficult problems, to call in the local wizard. Also needed are more flexible distributed management systems for large inter-networks such as hierarchical manager configurations. SNMPv2 offers greater capability in this regard by enhancing manager to manager communication. In the first case the blackboard architecture is a way of combining diagnostic reasoning modules with the usual SNMP or CMIP management functions. In the second case it provides a flexible way of building efficient combined manager/agent stations as might occur in mid-level managers.

In the sequel we first review the ways agents and agent/managers are used, the requirements the usage puts on them, and several different architectural approaches. We then discuss the blackboard architecture which has the advantage of being able to combining most of the best features of the individual approaches.

1.2: Agent Usage

Network management agents are organized primarily to act as a servers in client-server relationships with the manager entity acting as the client. In the server role the agent responds to requests from the manager. This relationship is most pronounced in SNMP but the model is relaxed somewhat through the agent use of traps whereby limited autonomous agent behaviour is permitted. Agent use in the OSI/CMIP domain is still primarily that of a server however more variety in usage is often called for. In addition to scoping and filtering demands CMIP agents may be called upon to create new conceptual objects at the request of a manager. In SNMPv2 too the agent role is also becoming more

varied because dual role manager/agents behaving sometimes as an agent and sometimes in a managerial capacity are facilitated through manager to manager communication to permit more flexible distributed network management models.

In addition to the simple conventional use of agents they are often required to perform in a proxy capacity. There are several reasons for this use. The first is that the device being managed does not support the network management protocol in use and so a proxy agent which does understand the management protocol and the device's foreign communication protocol is needed to intervene. Such proxies are needed when the managed device is too simple, such as multiplexors and simple bridges, or when multiple management protocols are in use. Examples of the latter coexistence case where the proxy is needed to translate occurs when an SNMP device needs to be managed from a CMIP agent or when a new SNMPv2 only manager needs to manage an existing SNMP agent. Often proxy needs are implemented through subagents who extend the capabilities of a standard agent and communicate with it.

Another use for proxy agents or subagents is to offload the device being managed. It is SNMP's philosophy in particular to make agents as lean as possible in recognition of the fact that the prime role of the device is usual to provide communications infrastructure and network management is a secondary role.

Another role of the agent may be to monitor several devices and so it must be capable of multiplexing managers commands.

Finally agents may be used as interfaces to other data base management systems. For example, routers have slots for interface cards and, while data on the kinds of cards the slot can take would reside on the router itself, a manager may want to query an inventory database about the availability and capabilities of various candidate cards.

1.3: Manager/agent usage

The mid-level manager sometimes acts in an manager capacity when it acquires information by querying various MIBs like the RMON MIB[8] or others. At the least, the mid-level manager would most likely filter information, format it, and send it on to a higher level manager for display and correlation. More desirably it could actually perform local diagnosis and pass on hypotheses or possibly more conclusive diagnosis. An interesting scenario is one where a manager requests data which is to be sent to a separate application for event logging and/or possible processing. In its agent role it might have to respond to requests for information from higher level managers or perhaps peers as might occur when there are different managers for different network views.

1.4: Implementation Requirements

First of all agents need to be compact and efficient. The extended uses for agents in SNMP and the demands on conforming OSI agents could require off-loading work to proxies or "agent servers"

Agents out of necessity must run on a wide variety of network devices and under a variety of operating systems. "Universal" agents structured as far as possible to be generic and to be movable are obviously desirable.

Agents must be available as much as possible.
Agents should be more reliable than the network.
Agents should not block when satisfying managers' requests and often have the need for multithread capability.

Agents and manager/agents often need to be run-time extensible. Networks are very dynamic and new devices and protocols are continually being added. The need for new information therefore arises continually and this takes the form of MIB addition or extension, depending on the granularity required. It may be necessary to add only one new variable to an agent's database or it may be necessary to add an entire new MIB, subtree, or MIB view. What is desirable or possibly necessary is that an agent be updated to accommodate the addition without having to interrupt its execution or to recompile it. At the same time the program's integrity must be maintained throughout the update process.

1.5: Achieving Extensibility

There are three basic methods for achieving extensibility and constructing subagents or proxies:

1. At the PDU level using the community string or context in the PDU wrapper to identify destination.

2. Using protocol methods such as SMUX or DPI to attach subagents.

3. Using the API approach whereby subagents write to standard interfaces or use shared memory.

The first method is quite simple and can be very useful in certain applications. Many organisations makes effective use of this method in its foreign proxy agent implementation. However it would be very difficult to implement SNMPv2 security mechanisms within this framework and stick to the formal protocol specifications. It is an ad hoc method.

The second method is through the use of a subagent which communicates with the agent via some special protocol. Subagents must register their capabilities, such as their MIB view, with the

agent or have this information built in with configuration files. Then the agent decides which subagent to use to field a query by determining which MIB or MIB view is involved. When a query requires the use of more than one subagent the situation becomes much more complicated depending on how requests are dispatched and with what granularity. Sets across subagent boundaries are particularly nasty to deal with. This approach is similar to the proxy idea, but in proxies the subagent is used more to overlay the MIB rather than extend it. Examples of the protocol approach include the SMUX and DPI protocols[3]. The SMUX protocol[2] has been deprecated but the DPI protocol is current. A new version has just been introduced and the extent to which SNMPv2 mechanisms have been accommodated is not clear.

Finally there are API or programmatic techniques, which to achieve dynamic extensibility would involve techniques such as dynamic linking, on-the-fly program updating, the use of RPCs, or the use of shared or overlaid memory in some way for extending agents. Some of these methods permit distributed applications where the subagent resides on a remote machine; others do not. Most often the agent activity is restricted to one machine but there are important cases where distributed activity is required. Regardless of the technique used for extension the clients of the agent should be unaware of the use of such methods. This is especially difficult to maintain if the client is a manager communicating with other managers using the SNMPv2 M2M MIB.

The more important techniques for agent extension are reviewed in a bit more detail in the next sections leading up to the blackboard approach. Only the most pertinent aspects are discussed here. The first three mentioned below are called external subagent implementation methods because the subagents exist as separate processes that communicate with a core agent using some form of IPC.

1.5.1: Community String Multiplexing

The SNMP community string can be used by an agent to multiplex requests to one of many (possible) proxies. This has been used where one agent monitors a number of simple bridges. Typically a configuration file is kept to list the various community strings and their corresponding executable proxy. In the UNIX environment a simple way of setting up agent to proxy communication is for the agent to fork and set up two pipes to the child who execs the proxy subagent. The proxy inherits the pipes. This is not the most efficient way of accomplishing communication but it is simple. Another disadvantage is that the agent can block writing requests to a busy proxy. The agent can decode the SNMP PDUs or it can pass the PDU on to the proxy to do it. If there are many proxy agents there is much duplication of code if each has to understand SNMP. An advantage of the method is that new proxies can be added easily with the limit being the number of pipes that can be open at one

time.

1.5.2: The SMUX Protocol

The SMUX protocol calls for the establishment of subagents called SMUX peers which are almost full SNMP agents in their own right. That is the subagent needs to understand ASN.1 and process SNMP PDUs, thus each subagent is fairly complicated. The subagent process communicates with the agent and registers its MIB subtree using the SMUX protocol. For each query the agent checks its registry to determine which subagent to pass the PDU through to. The mechanism uses basically course grained dispatching. In some cases however the agent may have to break the PDU into several separate request PDUs before forwarding to one or several SMUX peers. The agent will then have to correlate the results before sending a result back to the manager. SET operations can be complicated because of the requirement that they be all or nothing. It is best for the originating manager to insure that SETs involving more than one SMUX peer be issued with separate PDUs each destined for one of the SMUX peers, however the manager is not supposed to need to know about the agent architecture. Similarly for traps the subagent forms the packet which is passed to the manager through the agent. The subagent either gathers the needed data on demand or uses a cache ahead approach whereby the management base in its view is assembled locally and is kept up to data.

The method overcomes the limitations of static configuration from a configuration file and facilitates dynamic registration. Agent to peer communication calls for a connection oriented transport domain like TCP. In Unix this could limit the number of SMUX peers registered at any one time because there is a limit to the number of TCP sockets open at any time.

1.5.3: The DPI protocol

In the DPI approach the subagent uses a stream connection, typically TCP socket communication, to communicate with the agent and registers the variable or a branch in the MIB tree it supports with the agent using the light weight mechanism provided by the DPI protocol. The basic registration method is fine grained and is easily extended. The agent, on receipt of an SNMP manager request packet, authenticates and parses the packet and on a per varbind basis issues requests to the appropriate subagent using the simple DPI packet format. The agent is responsible for serializing and correlating the response into an SNMP response packet. The DPI protocol does not support atomic sets. For traps, the subagent issues a DPI trap packet which the agent converts to SNMP and passes on. Since the agents don't need to know ASN.1 or about SNMP PDUs, they need not be as complex as in the SMUX approach. The DPI protocol allows the subagents to reside on remote machines, a useful but not dominant feature. To get around the socket

limitation it is possible to modify the approach to use connectionless transport mechanisms like UDP.

1.5.4: API and Programmatic Agent Extension Methods

Use of Scripts

The HP "Extensible" OpenView Agent allows for the dynamic addition of MIB variables. To accomplish this the agent forks a process to exec a shell script to access an extended MIB variable. This is done for every added variable so the method is most important and useful when the need is to quickly add a variable or two. The method doesn't support tables however, and cannot be considered to provide a truly extensible agent.

Dynamic Update Methods

Systems for dynamic updating or on-the-fly program modification are attempts to address the problem of repairing and maintaining software as it runs. These don't appear to be useful for agents because of the overhead involved.

Use of Dynamic Linking/Late Binding

A cousin of dynamic updating but somewhat more limited and less robust, dynamic linking is used mainly for updating shared libraries but can be used to extend and update agents on those architectures permitting its use. It is reported to be popular in some commercial agents but the exact implementation is operating system dependent. The method amounts to dynamically replacing or adding procedures or functions to handle the new MIB variables. As such the method can be described as a procedure call approach in which the entire agent operates as one process. This important method is proposed for the Open SNMP Agent for which the specifications are available[]. A very closely related specification for a Windows SNMP Agent calls for almost identical functionality but for implementation of subagents as "external" separate processes communicating using shared memory messaging for IPC.

Use of RPC

Here the manager invokes remote agents via RPC calls. This is fairly complex to set up but DCE for example has tools to facilitate RPC through the IDL language compiled to stubs. The RPC approach is not so much a full fledged management philosophy but is rather a collection of tools that are applicable. DCE/RPC has the advantage of DCE's multithread capability that solves the blocking problem that can occur in other approaches. We note that DME on the other hand is a user interface or API that allows for use below of either RPC, SNMP or CMIP methods.

AS we shall see, the blackboard architecture for agent or manager/agent implementation is a similar but more comprehensive shared memory method that provides a standard API, the blackboard handling functions, for subagents (KSs) to interface to the blackboard with.

The DMI Specification

The DMI Specification[6] describes for desktop management a structure that has an overlay area for dynamically executing code. The specification deals with only the low level physical device side of agents, but it allows the agent to use any one of a number of sets of code at any particular time. The DMI model uses a representation for data called the MIF format which is very rigid as compared with MIBs and is restricted in size. The mechanism for invoking code involves the use of a MIB browser which can look up MIFs on disk. MIFs have a line which can give the path to dynamically executable agent code which is then put in the overlay area. This is a procedure call approach.

2: The Pegasus Blackboard

The Pegasus blackboard has four major components. The knowledge sources (KSs) are a group of processes which work together to diagnose network faults. The blackboard database is used by the KSs for interaction and communication. The blackboard handling

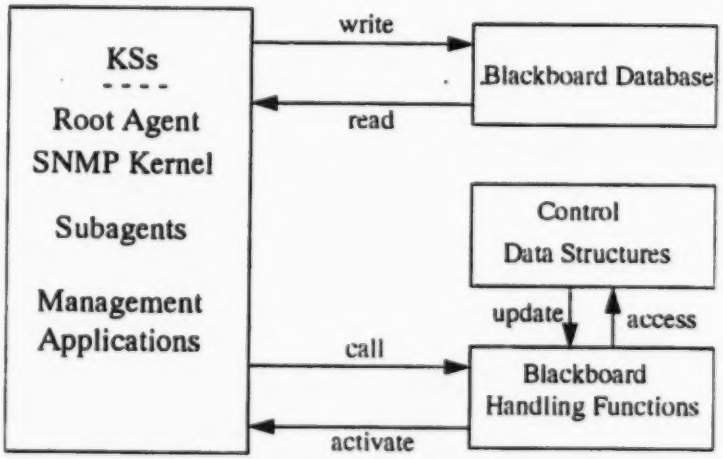

Figure 1
Pegasus Blackboard Architecture

functions provide the KSs with blackboard access. The control data structures facilitate process communications and permit blackboard data management. KSs can be written in several different languages, C, C++, Pascal and Prolog have been used so far. The blackboard control functions are written in C and C compatible C++.

The blackboard database is implemented as shared memory, holding the most current data about the network, and as a sequence of history files, recording previous time periods of the blackboard. Processes access the shared memory (current data) using the same logic as they access historical data. The current blackboard database is partitioned into seven areas to improve access speed.

2.1: Blackboard control

Control data structures support the management of the blackboard. Apart from the usual lists of free records and mutual exclusion semaphores there is a KS registry structure and structures which partition and manage the blackboard partitions.

The KS registry contains the names of KSs presently connected to the blackboard and a pointer to a control block for each such KS. Software processes convert to KSs when they register on the blackboard using the functions provided for that purpose. At that point other KSs can detect their presence and so can make use of any facilities that are known to be offered. KSs can deregister and reregister freely. This means that processes can start, be debugged, stopped, edited, recompiled and restarted without altering the blackboard or other operations of other KSs. This formal step of coupling processes to the blackboard to change them into KSs is crucial to the runtime durability of network diagnostic systems built using this method.

KSs typically wait for some change of interest to occur on the blackboard. The change means that information has been put on the blackboard that this KS may be concerned with. The blackboard has wait functions which allow the KSs to perform blocked or unblocked waits for changes on one or more areas. The private semaphore for a KS is triggered by a change in a specific area. A set of structures support the awakening of KS when something changes in an area they are interested in. This restriction of a KS to waiting for a specific area to change greatly reduces the time KSs spend checking to see whether or not they should act. Three major components exist for each area: a list of KSs interested in this area, a list of recent changes to the area and a private semaphore for each KS.

2.2: The blackboard database.

The blackboard is partitioned into seven areas to improve

access speed. The seven areas, described briefly below, are: status messages, traffic messages, problems, requests, replies, hypotheses and KS activation records.

Network monitoring systems produce status and alert messages. Their formats vary from time to time, depending on the monitoring systems used in the network. Their rate of flow can range from a few per minute to many thousands a minute, depending on the network and its state.

Some network monitors produce traffic statistics about devices (eg: a line's transmission and error counts over the last minute) Again the formats vary from time to time, depending on the versions of systems installed. The flow of such messages is usually very constant, averaging many thousands a minute for a large network.

All network problems are reported in the problems area. Such reports include telephone messages derived from user complaints to network operators. They also include problems spotted by diagnostic KSs.

Any KS can ask for another KS to perform some service. The client KS writes a request into the blackboard, using the appropriate record structure for that request, filling in the record's fields as if it were a job request form. The server, which is programmed to respond to that request, reads the request. It will carry out the action if it can and will reply accordingly, Such a request might be to attempt to login to a network node to check that it is active. The reply in such a case would indicate whether or not the login was successful. Other common requests are for SNMP actions, the replies then being either the SNMP requested information or a SNMP reply timeout status.

The diagnostic KSs (processes) typically wait for some information to appear. For example the BANES break fault diagnostician reads each status message as it appears, waiting for the next one once it has digested the information from the previous message. Periodically BANES considers the information as a whole and will report hypotheses about devices being broken. Other KSs wait for such hypotheses and will then try to confirm or deny them. For example, one KS will attempt to login to nodes that BANES reports are broken. Nodes that are hypothesised as being broken and which cannot be logged into are then reported to the problem area.

Some KSs need to control several others (the master:slave relationship). KSARs allow a KS to enable and disable the operation of selected KSs. This allows for a master KS to deliberately execute a planned series of slave KS operations in order to carry out a familiar task. This mode of operation is necessary for efficient execution of routine, multi-step tasks. It also allows for much more complex sequences of operations. For example, one KS (NPATH) will, given any two node names, determine the 15 most

likely paths between them. At one time this KS can be slaved to another KS which is trying to track down a file transfer problem. At another time, the NPATH KS might be slaved to a KS which is checking a hypothesis that a router has a routing logic error.

2.3: Blackboard tests

The blackboard has now carried several hundreds of millions of messages and several tens of gigabytes without problems. It has also been tested and operates correctly with several hundred simultaneously active KSs, although is usually used with only a few active.

When a message is passed from a KS onto the blackboard and back to another KS, the CPU time taken to do so is less than if the message were passed through a mediating process or the operating system. This is due to the fact that the blackboard is a shared memory segment. Hence although extra overheads in data indirection and control are met, these are less than those incurred by conventional process management (at least in the SUN4 uniprocessor that was used). On this machine each message took 0.0012 seconds of cpu for the entire creation, transmission via the blackboard and receipt process.

2.4: KS Organization.

KSs can be distributed. Suppose that a KS is to be moved to a machine other than that which contains the blackboard. This KS is divided into two parts. One part contains all the KS logic that deals with interfacing to the blackboard. Since this part is usually very small it is termed a mini-KS. The other part, which carries out the KS's primary actions, is reconstructed as a separate process running in the other machine. Two-way communications (eg: UNIX pipes or sockets) are now established between the mini-KS and the new process. The mini-KS runs in the same machine as the blackboard and communicates to and fro with the rest of the original KS, now running as a separate process in the other machine. The blackboard is not aware of this.

This distributed KS technique has been tested by writing Prolog KSs running in different SUN workstations. These distributed KSs each had their own mini-KS running in the workstation where the blackboard ran. Communications were run across the LAN that connects the SUNs. At the same time, two other KSs ran conventionally in the same machine as the blackboard and interactions between all the different KSs were shown to work correctly. Since many hundreds of KSs can run at once, this technique means that the blackboard can be used to control massively complex diagnostic systems, even though the cpu on any one machine would be insufficient to handle the load.

Over a dozen KSs have now been developed. These include the network simulator (DRAGON), the break fault analyzer (BANES), a neural network detector of specific router performance faults (LEAPR), several expert systems for verification of faults, the NPATH KS mentioned above and a multi-level diagnostic reasoner implemented in a distributed manner. A case based reasoner and two machine learning methods are presently under KS development. The very strong modularity imposed by the blackboard architecture means that the developers of each KS need only understand the services provided by two or three other KSs. This means the complexity of programming a KS barely changes as the number of KSs increases. In other words, the complexity of programming increases much less than linearly with the total size of the system.

The job of writing new KSs has proven to be unexpectedly easy, for two reasons. First the particular choice of the seven areas for the blackboard has reduced the difficulty. Since programmers only need code their KSs to scan changes in the area they need to be familiar with (eg: status messages), programmers need not learn all the details about other areas. Second, many KSs are very similar. Most KSs wait for changes in an area, make simple requests of other KSs and wait for and read replies. New KSs are now invariably designed and coded by changing existing designs and code. This means that quite inexperienced programmers can produce useful and effective code in what, in other architectures, can be an exceptionally complex and challenging real time system.

KSs can be organized to cooperate as peers or to work in hierarchies or to work in more complex groups. The hierarchical view has been employed where one KS acts as a service agent on demand for other KSs. For example, NPATH is used by other KSs in order to verify router logic and to search for file transfer problems in the most effective manner. However, this view of NPATH as playing this role for these other KSs does not prevent it being used in other roles in future. In other words, the system designer who uses a blackboard and KSs can use one set of views about KS roles to solve one problem and another set of views of the same KSs to solve another. The KSs do not need to change to permit these differing views. This flexibility is very hard to achieve in conventional multi-process systems, and usually at severe cost in reliability or complexity.

2.5: Summary

The interprocess functionality has been efficiently achieved with very solid reliability. Four exceptional and unexpected benefits have been noticed. First, since KSs are so modular, so isolated from each other and so easily tested and debugged, inexperienced programmers have been able to rapidly produce operational KSs.

Second, the difficulty in adding new KSs does not increase with the number of KSs, it remains almost constant. Thirdly, the isolation of the network from the KSs and the provision of so many data base features in the blackboard has encouraged the production of diagnostic methods that are easily tested against other domains. Finally, the rugged nature of the blackboard ensures that all bugs are due to each new KS and practically always only affect the new KS. This means that no maintenance support is needed by the blackboard.

2.6: Evolution to a Blackboard Implemented Manager/agent

It is the KS features of the blackboard that provides much of the functionality needed for agents and agent/managers. It makes it possible to construct a mid-level manager in a very modular way by simply adding KSs for:

1. SNMP serialization and BER encoding or decoding. Such a KS would act in the role of an SNMP kernel, receiving request packets, and formulating response packets. Actual work would be farmed out to a KS acting as a root agent or to a KS acting as a managing subagent depending on which capabilities are required from this dual rolled midlevel element.

2. Subagents. Addition of new MIBs and MIB views dynamically is accomplished by subagent registration with the blackboard using the registration functions similar to those for ordinary KSs (the API). Dynamic extensibility is thus provided. Subagents need not know ASN.1.

3. Dispatching. A KS to parse, control, schedule, and dispatch requests associated with Mib variables will be added to act as a root agent. Dispatching to subagents will be on a fine grained per varbind basis much like in the DPI protocol. It is with the root agent that subagents actually register. The root agent will also coordinate replies from several subagents to provide a unified response to a request. The root agent will also receive and forward trap messages being sent by subagents. At this level a light weight protocol and not SNMP is used.

4. Management applications. KSs to perform midlevel management operations such as data aggregation and region diagnosis can be added as managing subagents which support MIB variables and perform management functions.

5. Distribution. Subagents residing on remote machines can be accommodated through an additional KS. In this case the DPI protocol could be used for communication.

6. Coexistence. Other protocols like CMIP can be accommodated in a modular way through the use of KSs.

All KSs are activated by semaphore and have in them a defined core of management functions for plugging into the blackboard API.

2.6.1 Advantages

All the previous mentioned advantages of the Pegasus blackboard are carried over into the blackboard manager/agent. In addition:

1. All subagents (KSs) can register and deregister freely, hence support for a MIB view can be removed.

2. There can be a mixture of regular KSs for diagnosis and other applications and manger/agent KSs. These can all communicate and interact in an integrated environment.

3. Message creation, transmission, and reception is very efficient through the use of semaphores.

4. KSs are very loosely coupled, making for a more manageable concurrent system.

5. The architecture offers flexibility for OSI implementations where scoping and filtering need to be optimally placed.

6. Agents do not have to block waiting for subagents.

7. There is, we believe, a certain elegance and symmetry to the solution.

3: Service Management Architecture

The example network environment of the appendix is quite representative of a broad class of heterogeneous network configurations which span multiple subnetworks and possibly a number of separate administrative domains. Here we have the operators from the A and Q networks as well as from the carrier D. The problem arises of deciding what information should be gathered from the networks and jurisdictions to facilitate management in general and diagnosis in particular. Once the case is made for the gathering of various data, and assuming that one can get agreement from the carrier for example to provide certain data on a regular basis, there remains the question of how the transmission and delivery should be accomplished given the multiple protocols occurring in the heterogeneous environment.

In order to decide on the appropriate data, one needs to make decisions as to the possible diagnostic algorithms and the cost/benefit ratio of each item. The proposed complexity analysis is an effort to come to grips with the relative comparison and tradeoffs that would influence the design of the management data

acquisition system. Its goal is to help focus attention on network problem areas, to help determine the classes of faults that need to be addressed and what information needs to be provided for appropriate management. There are two focal points here, the physical network decisions that affect the diagnostic complexity and thereby the diagnosibility of the network, and then there is the management information which also must be chosen to influence the diagnosibility.

Once these have been chosen the information must be modelled in such a way as to proved interoperability across the network. For example A must be able to understand information from carrier D and perhaps also from A and for this they need a common data model. The model then can be used as the basis for a management architecture.

Given the current state of development of network management schemes and protocols, network operators traditionally tend to regard their view of the network as a collection of components. From the management protocol design point of view there are several approaches to facilitate management of heterogeneous networks. For current and future networks including BISDN the issue seems to be focussing on the management of multimedia data streams and services from end-user to end-user over several different customer premises and public network domains. Current thought seems to centre on the management of services across a number of separate administrative domains as opposed to management of devices which is the more traditional approach. This requires access to management information within each of these domains which are administered by separate organisations, consist of equipment provided by a number of different vendors, and supply a variety of different services. The task is to provide service users A and Q for example a single logical view so they can perceive the services as if they are provided across a single management domain. Then this view must be reconciled with the low level device oriented views. There are several efforts underway examining how to provide a service management architecture with an overall hierarchical management. structure. How this affects complexity and diagnosibility has not been addressed in these initial investigations. The single logical view surely reduces the complexity from point of view of A but makes the job of D more complicated to accommodate A.

3.1: The Internet Approach

The internet approach is based on recommendations of the IETF Interface MIB working group.

3.1.1: The Service Management Architecture

The SNMP based approach concentrates on the development of a Service MIB Module (as opposed to classical device MIBs) as a basis for a more comprehensive Service Management Architecture which differs from a device management architecture. The Service MIB provides to customers a logical view of the customer's portion of

a shared network by modelling the service not the underlying implementation or devices. The service network is modelled as a single entity or logical device. Physical customer connections to perhaps multiple switches or routers are modelled for the customer as individual interfaces on a single logical device. To answer a customers SNMP query, a Proxy agent must map the logical view back to the physical and then queries the low level devices in the traditional way. Similarly in the other direction an SNMP trap originating at a device must be mapped back into the logical view.

Customers may of course be using multiple service providers' networks, Bell,Unitel etc. as in DND's case. The customers end-to-end view consists of multiple segment end-to-end connections that span across multiple service providers' networks. Each service mib concentrates only on an individual service provider so the enterprise Network management station (NMS) has the job of consolidating the individual service provider segments into a unified customer end view. How this should be done will require a lot of future work and experience.

3.1.2: The SNMP Service MIB

A lot of thought has been given already to the service MIB. A service provider may offer several network services such as Frame Relay, ATM etc., and may facilitate interworking of services by providing for example virtual connections between frame relay and ATM ports. To accommodate all this the service mib provides a single generic view of network configuration including information on logical ports, access channels, virtual connections etc., regardless of network service or underlying data link protocol. These are provided in several proposed new interface tables. The mib also provides mib modules specific to a given network service or datalink protocol. These contain tables which are connected to the "generic" tables via indexing or object id pointers. Thus a hierarchical relationship between the "generic" and "specific" layers is established.

3.2: An OSI Approach

Similar work is also going on in the OSI world as for example in the RACE project. The RACE OSI effort is directed toward integrated broadband environments in which end users will be provided with a variety of different communications services for voice, data, and video over a single network. This also involves monitoring and control of end-to-end data streams over several different management domains including Customer Premises networks(CPNs) and Public Networks (PNs). In order to structure and design the required inter-domain interactions between management systems a large effort to thoroughly define all the appropriate management services, information models, and interoperable interfaces was undertaken. This was done by first defining the management services and mapping them to managed objects in the

supporting information models.

3.2.1 Management Services

As usual, in accordance with CCITT/OSI recommendations, the approach is strictly layered in which each layer provides management services to the layer above it. The structuring was accomplished by dividing the network into three layers the CPN layer, the VPN (Virtual Private Network)layer and the lowest level the MAN or WAN layer. At the lower level are the network providers who in addition to the WAN and MAN service operations support functions (OSs) , provide management services to a value added service provider (in this case a virtual private network (VPN) service provider) positioned at the layer above. The VPN service provider adds value to these services and provides services to the layer above. At the top are the operational support functions (OSs) of the CPN. In this model also, the services provided by the network providers are not directly visible to the CPNs OSs however the services made available to them by the VPN service providers may map to services provided by the network providers. For example the CPN manager may specify a request which the VPN service provider must map down to the MAN or WAN information model. The VPN service provider then uses a management service provided by the WAN OS. These services support the VPN service by providing the VPN service provider with the capability of mapping its own services onto the WAN bearer services. As in the Internet approach, the orientation of this work is to providing service modelling as opposed to device modelling. The services are described using the "Abstract Service Definition Convention"

Service management capabilities provided to the CPN OSs (inter-domain end-to-end management services) are defined by the services provided at named ports by the VPN service provider. The latter includes

1. customer administration services
2. end user access management services
3. traffic and switching management services supporting creation of connections between service end-users
4. quality of service information and notification

Needless to say the QoS services are pretty well limited to notification and do not include diagnosis.

Management services for VPN support take the service management functionality offered by the network providers and incorporate them to provide an integrated view of the VPN service across heterogeneous networks. Considering the nature of a VPN from the service provider's point of view, the service provider views the network in terms of entities which will enable him to set up and run VPNs. These are service user groups who share the same VPN and

linkages which allow the service provider to create dynamic linkages between existing service groups to form VPNs. Although at first glance the Internet and OSI approaches appear different from the implementation point of view, they really are quite similar in intent.

3.3 The ODP Approach

The Open Distributed Processing framework[13] which is getting a lot of attention these days, presents an interesting approach to service management. In the framework a service is defined to be a set of functions offered to a user by an organization. Such an organization is said to be an exporter and a user is an importer. Services can be offered by more than one exporter. In ODP, service management can be organized based on an architecture of objects called traders. Exporter objects offer their services to traders who act as mediators. Profiles of each announced service are stored in the Service Directory of an ODP trader. An importer looking for service gives a defined service description (a service request) to a trader who searches his directory for a suitable service. If a trader can not find a suitable service for an importer request it can contact other traders via an administrator.

For network management, ODP trader architecture objects can be mapped onto OSI type managed objects which can be monitored and/or changed by a manager. Managers can also be managed objects for higher level managers. An importer request to a trader is a user request to monitor exporters or to choose one of them. ODP service objects can be represented by managed objects whose attributes are related to the components of the service directory.

Large networks can be divided into domains (possibly overlapping). A domain permits a group of managed objects to be treated as a single entity for the purpose of specifying policy, which provides the basis for coping with the complexity of large scale distributed systems. Management authority over a group of managed objects can be delegated by one manger to another.

It is interesting to note that recently the Pegasus team at Carleton experimented with service and resource management using a similar architecture[12]. The interesting innovation was that services were managed through a competitive bidding process of "trader" like processes. This scheme was being explored as a vehicle for managing future systems with dynamic service capabilities.

3.4 Summary

These developments and customer demand are generating pressure on the providers to provide (CNM) Customer Network Management of network services over shared networks. The purpose is to give customers a management window into their portion of shared

networks. None of the development so far has concentrated on what statistics should be associated with the logical components and how they can be incorporated into a hierarchical diagnostic framework. Some work has been done in the T1M1 area on standardizing event reporting across jurisdictions but this has been completely device oriented and offers little to the newer service oriented architecture. Just recently the Network Manager Forum proposed a trouble ticket standard for sharing real-time network status and trouble information between domains in a common format. It is not clear yet where this fits into the service management scheme of things.

In summary also we emphasize the need for Local Diagnosis and Distributed Management The current defacto standard (SNMPv1) is relatively easy to use to develop management applications but overhead in data transmission is high. Remote discovery is expensive. There is no provision in the protocol for intelligent filtering at the source and no provision for peer-to-peer communication (it is based on the pure client-server model) so it not suitable in itself for distributed hierarchical management.

Proliferation of subnetwork types and protocols is making it more and more difficult to gather together at one NOC the kind of expertise needed for centralized management. Much of the current diagnostic capability can only be performed locally. i.e. Sniffer type protocol analysis on a LAN segment. This approach is very expensive and requires promiscuous monitoring.

For these reasons the remote monitoring MIB (RMON MIB) and later SNMPv2 with its manager-to manager MIB were introduced. SNMPv2 now provides for midlevel managers within the management protocol however the use of the manager-to-manager MIB still appears to be limiting and quite complex so it is not widely deployed.

E. CONCLUSIONS

1: The origins of issues.

The issues in the management of heterogenous computer networks operated in multiple, variable, layered domains rise from a common source - uncertainty. The issues are to be tackled by first reducing this uncertainty and then managing the residue. Uncertainty in these networks is not created, it exists anywhere operators have too little knowledge. Networks with the attributes mentioned require a generally higher level of knowledge to operate than simpler networks. This is because simpler networks (for example homogenous ones) allow operators to interpolate knowledge safely (since the devices are similar). More complex networks require knowledge to be gathered about each complication. The management issues arise from how to know where the knowledge may be weak, how to gather the knowledge and how to use it.

2: Where knowledge is usually weak.

The points in a network where two management domains overlap or touch are well known to be dangerous. The operator of one domain has invisible elements that connect portions of the domain together. The invisible elements are operated by another domain manager. These invisible elements are the source of much of the danger and complexity. The points at which they connect to the visible domain are the points where management of knowledge can greatly reduce the danger and complexity. The general principle involved is to increase the amount of knowledge at these points in the network. The knowledge is of two types, knowing that the components which are connecting are correctly configured and knowing the historically acceptable performance attributes of the invisible elements as a whole. This general principle derives from the need to reduce network diagnostic complexity at points where such complexity carries the highest potential cost. The principle translates into two policies as follows. A device connected directly or at one remove from an invisible network element must have its configuration double checked by someone other than the installer. This means that a router connected to another network must have its configuration double checked. The second policy is as follows. Invisible network elements must have their entry and exit traffic and error levels recorded so that current levels can be readily compared. This allows the operator to readily determine if the invisible element has significantly changed its behaviour and provides the means to convince the operators of the invisible element that they have a problem.

The other places in a network where extra density of knowledge is desirable are of course core traffic areas, places which have

been changed recently and places where problems have occurred before. Routinely managers focus of the core traffic locations, but a great many problems occur when changes to networks are made. Therefore a responsive monitoring system that reacted to change by concentrating on gathering more information from and near locations of change would reduce the difficulty of diagnosis of problems related to change. Finally, managers seldom respond to places where previous network problems have occurred by raising monitoring levels, rather the problem is assumed to have been solved. In reality, problems are often only superficially solved and reappear again and again. The later incarnations of the problem often become fiendishly difficult to diagnose. A way to anticipate and reduce the severe cost of these long standing problems is to assume that where a problem has occurred before, another is likely to occur again.

3: How to gather the knowledge.

Within the framework of current network management protocols, the basic monitoring model is a client - server one with the centralized manager acting as client and the agents as servers. To be flexible and to be able to adapt to new and diverse services, devices, and management applications it is necessary for the management system to be extensible. Most of this load falls on the agents and hence techniques for building extensible agents are particularly important.

The pure client - server model is sufficient for only relatively small networks because as the network size increases the volume of management traffic across the wide area network becomes excessive. Large networks invariably resort to a distributed management scheme (usually hierarchical) in which distributed midlevel managers bring managerial intelligence out into the network and leave only the most important high level issues for the central top level manager. The midlevel managers are dual role, sometimes serving as a server/agent to the central manager and sometimes serving as a local manager to a group of agents situated on critical network elements. To accommodate evolution of the network, the midlevel managers also need to be extensible And this places additional burdens on extensibility architectures. The Pegasus blackboard architecture appears to be a particularly viable way of implementing such management components.

4: How to use the knowledge.

Network operations centres usually have wonderful graphics, superb remote access methods and connections to state of the art

monitor and line level diagnostic systems. They are usually very poorly equipped to perform statistical analysis of any kind, because they perceive this as seldom being useful. Indeed, it is seldom useful. However, proactive management and management to reduce the difficulty of the very hard diagnostic problems requires routine statistical operations to be carried out. Only when managers perceive the extra day to day cost of such operations to be worth while in reducing the likelihood and cost of severe faults will they be undertaken. However, since managers often are changed in NOCs, they are not often in place long enough to perceive the tradeoff. In some ways this requires a change in view by the personnel. Rather than seeing themselves as firefighters, they should see themselves as fire preventers.

F. REFERENCES

1. Pagurek B., Dawes N., and Kaye R., " A Multiple Paradigm Diagnostic System for Wide Area Communications Networks" IEA/AIE-92 5th International Conference on Industrial and Engineering Applications of AI and Expert Systems, Paderborn, June 1992.
2. Rose M., "SNMP Mux Protocol and MIB", Internet RFC 1227, May 1991.
3. Wijnen B., Carpenter G., Curran K., Sehgal A., and Waters G., "Simple Network Management Protocol Distributed Protocol Interface Version 2.0", Internet RFC 1592, March 1994.
4. Romanov A., White W.H., and Wilson P., "The Open SNMP Agent" v 0.2 Paul Freeman Associates, March 28 1994. Available via ftp from ftp.uu.net : tmp/openagent.txt.
5. Romanov A., White W.H., and Wilson P., "The Windows SNMP Agent" v 0.1 Jan 1 1994. Available via ftp from sunsite.unc.edu : /pub/micro/pc-stuff/ms-windows/winsnmp
6. "Desktop Management Interface Specification" Version 1.0, April 1994. Available via ftp from aurora.intel.com : /pub/DMTF.
7. Case J., and Levi D., "SNMP Script Language" Internet Draft draft-levi-snmp-script-language-00.txt, October 1993.
8. Waldbusser S. "Remote Network Monitoring Management Information Base" Internet RFC 1271, Nov. 1991..
9. Case J., McCloghrie K., Rose M., and Waldbusser S. "Manager-to-manger Management Information Base" Internet RFC 1451, April 1993.
10. Iqneibi S. M., "A Blackboard Architecture to Support Network Fault Diagnosis" M.Eng Thesis, Dept. of Systems and Computer Engineering, Carleton University, May 26, 1992. Published as Departmental Report SCE-92-11.
11. Dawes N.W., Pagurek B., Iqneibi S., and Yuan B., "An Effective Architecture for Communication Network Diagnostic Systems" Avignon '93, 13th International Conference on AI, Expert Systems, and Natural languages, Avignon, June 1993.
12. Dawes N.W. and Kalab P., "Competitive Software Agents Providing Communications Services" Carleton University, Department of Systems and Computer Engineering Report SCE-93-20, 1993.
13. ISO/IEC JTC1/SC21/WG7 reports X.901, X.903, 1992.

G. APPENDIX 1

Examples of practical problems.

Let D operate a network spanning Ottawa,Toronto,Montreal and Winnipeg. This network is used by many companies. Company Q leases a fixed capacity from D on this network and retails portions of the capacity on a demand basis. In other words D is the wholesaler and

Q is the retailer. The customer, named A, has a large WAN. However, from time to time the WAN core (spanning Ottawa, Toronto and Montreal) runs short on communications capacity. At these times the network manager buys in capacity from the Q company and ties it into the routers at each of the three core sites. In other words, Q provides line capacity between data routers in Ottawa, Toronto and Montreal. A, the customer, owns the routers. Q owns the communications line connections and leases the communications capacity. D owns the muxes in each city and the fibre running between the cities.

There are three levels of operation. D runs the physical links. D also runs the multiplexed channels on the physical links. Q runs the connections into the muxes. A runs a WAN which includes the Q and D components as layers. If the Ottawa - Toronto link fails in the WAN, who can diagnose it? The answer depends on the problem.

Problem 1:

Cut on the long haul fibre.

D knows about this immediately due to loss of transmission integrity checks at each end. D know to call Q, one of their customers on this line. Q may know to call A. The operators in the NOC of A should know to call Q, if they can isolate the problem. However, isolation of the failure by A is only possible by knowing all about the A and Q and D equipment topology and status. The operators in A therefore have serious problems and are probably dependant on the service response quality of the operator at Q who gets called by D, as well as depending on D to call Q.

Problem 2:

Erratic signal loss on the Q connection to the D MUX at Toronto.

This intermittent, short lived failure mode causes occasional automatic reset and restart in the line card in the Q connection to the D MUX. The failure is caused by erratic power levels due to an inadequately pushed in power connector. Significant vibration of the floor will cause the power to blip. This fault causes occasional very slow periods for transmission with consequential file transfer failures in the A WAN. In turn, this triggers occasional buffer overruns in the Ottawa router as it stacks packets to send to Toronto, due to misconfiguration of the router in Ottawa.

The users of A get occasional file transfer failures, but are used to them. The operators of A see that the Ottawa router performs ok, but they set the alarm levels up as it is often

telling them there are retries but no one is complaining so clearly the alarm levels were set too low. The company Q never see a problem, as the automatic reset and restart are not alarmed, as there is supposed to be full redundancy and only serious problems needing attention are alarmed to the Q NOC. D never sees a problem.

One day the traffic on the road beside the Q device with the poor line card rises dramatically, due to road works elsewhere in Toronto. The line card now resets and restarts faster and faster. After a few days of this it will fail hard, but on the first morning the restart cycle averages three minutes. In Ottawa the traffic rises through the morning and the Ottawa router starts to alarm frequently. The A operators never notice the slow rise in alarms as the Ottawa router is always a bit flaky. Finally a user starts a 1 Mbyte file transfer with 1Mbyte buffer size. This buffer cannot be transmitted to Toronto in the intervals between line card resets. The user complains to the A operators. They can send other files readily to Toronto. How can they diagnose this problem? They actually have three days to solve the problem before the failure becomes obvious and very serious, but would tend to write it off as user finger problems as it is not reproducible.

Problem 3:

Signal loss rate goes up when the long haul fibre is temporarily replaced by a microwave link.

Assume that in this example D are now prevented by CRTC regulation from telling company A that they (D) provide the wholesale service to Q. This might lead to D poaching Q's clients. Q are told in a written memo by D that the replacement will operate for a week, June 1-7th. Q do not inform the operators of the A WAN. However, on June 3rd the retry rate over the microwave line Toronto to Montreal rises by a factor of 10,000 while a thunderstorm moves along the St Lawrence. During this time the router in Ottawa gets overloaded by the consequential dynamic rerouting executed by the Montreal and Toronto routers, as they try to unload the error prone line. The Ottawa router crashes with memory overrun, due to its misconfiguration, and restarts with full alarms. The A operators get flooded by angry user calls as now Montreal, Ottawa and Toronto have practically lost contact. All the A operators see is the Ottawa router crashing and restarting every 15 to 20 seconds. They shut down some lines into it, to unload it, and as the users simply give up on the network, the system recovers. Postmortem analysis by the A management team leads them nowhere. However, the next day another thunderstorm goes through and the same catastrophe occurs. The problem is almost insoluble for A, right at the very highest level of analytic difficulty for experienced operators. Of course, should the A WAN have full router redundancy in Ottawa, the problem will become even harder to diagnose, but not much less likely to happen.

This problem is symmetric to a set of related problems which would be enormously complicated or rendered insoluble by security considerations.

Notes on the problems.

These problems have been constructed from combinations of real events.

Solutions.

The solutions to these problems are all of the same nature. The class of problems needs to be anticipated. Information must be gathered in advance to allow for simple resolution of problems in the class. The difficulty of problem resolution in the class needs to be reduced by taking very low cost preemptive measures. In other words, although particular problems cannot be solved in advance, management methods can be deployed in advance of the problems in order to make them easy to handle.

The general principle involved is to increase the amount of knowledge at points in the network which connect into unmonitored or invisible network elements. The knowledge is of two types, knowing that the components which are connecting are correctly configured and knowing the historically acceptable performance attributes of the invisible elements as a whole. This general principle derives from the need to reduce network diagnostic complexity at points where such complexity carries the highest potential cost. This almost always means reducing uncertainty in the network at exactly those places in the network where invisible elements intrude. These elements will be managed by others (eg: leased network segments, a dynamic and variable topology or a different layer). The principle translates into two policies as follows.

Policy 1:

A device connected directly or at one remove from an invisible network element must have its configuration double checked by someone other than the installer. This means that a router connected to another network must have its configuration double checked. Invisible network elements are those which the operators cannot directly monitor and usually are those which are managed by someone else.

Policy 2:

Invisible network elements must have their entry and exit traffic and error levels recorded so that current levels can be readily

compared. This allows the operator to readily determine if the invisible element has significantly changed its behaviour and provides the means to convince the operators of the invisible element that they have a problem. It also reduces the uncertainty associated with diagnosis at the interface and hence the complexity of such diagnosis. Finally, should statistical analysis be routinely carried out on the traffic and error data, anticipation and hence prevention of problems can occur.

Solution to problem 1.

When the link goes down, the operators in A know it because the traffic and error statistics for the invisible element unambiguously point that way. The A operators only have to check two things to find this problem, the invisible element statistics and that the devices connected to the invisible element are operational and have been double checked. The A operators now call the Q operators to complain of loss of service.

Solution to problem 2.

The double checking of the configuration of the Ottawa router should catch the anomalous setting of the retry error alarm levels. If this alarm level is set correctly the A operators will be complaining to the Q operators about the intermittent problem long before the device finally fails. If the double checking is not done correctly, the collection of the statistics about the invisible element will, if analyzed, show a systematic degradation in performance of this element. Operator analysis of the file transfer will now have enough information to locate the true problem as being in the Q operated link.

Solution to problem 3.

The misconfiguration of the Ottawa router should be detected when it is set up, by the double checking policy. If is not detected, the change in line by D will cause a change in the error statistics collected by A about the invisible element. Even if these are not analyzed before the system crash, they will clearly point to the problem during the postmortem. In other words, for this extremely difficult problem the two policies combine to reduce the likelihood of occurrence and to make it far easier to solve if it does occur.

H. APPENDIX 2 Complexity

Network fault prediction and its consequences.

If the probability of particular faults occurring at particular locations in a network were predictable, many consequences would follow. The network could be altered and the probabilities recomputed, until some minimum failure figure of merit was satisfied. The information gathered from the network could be selected to optimize fault diagnosis in order to improve the mean time to repair while minimising the information load. The primary result is that networks could be designed rather than just built. However, in order to provide for these happy circumstances, fault probabilities need to be predictable.

A measure has been sought that directly correlates with the mean time between failures (MTBF) of a device and is easily computable. This measure is called complexity.

Definitions:

Component: components are the software protocol modules and the physical line modules in a device.

Device: a device communicates with other devices using physical media.

Protocol: a layer of logic that defines how information shall be exchanged and interpreted to enable communication. A heterogenous network contains devices which have more than one protocol at a layer.

Variety of use: the number of different ways an object was used in the last T time. For example, if a data router handled packets addressed to 95 different target logical addresses, it would have variety 95. Each update of the router table this would use would have unit variety for each such update. A bidirectional line has variety 2, if used in both directions. A large voice switch which handled 17,800 different calls to different phone numbers would have had variety 17,800 for this function.

C_j = complexity of device j

${}_iC_j$ = complexity of component i in device j

m = the number of components in device j

N_i = the variety of use in component i in the last time period T.

P_i = a priori probability that component i will have a fault in time T per variety of use.

T = the number of minutes over which complexity is computed.

$${}_1C_j = \log(1 + P_i * N_i) \quad \text{..... 1.0}$$

$$C_j = {}_1C_j + {}_2C_j + {}_3C_j \ldots {}_mC_j \quad \text{.....1.1}$$

Examples.

1: A line is bidirectional and carries ATM traffic in and out of a TCP/IP : ATM gateway. Its P_i = 0.001 and, because it is bidirectional, it has variety of use equal to 2 .Its complexity is therefore log (1 + 0.001 * 2) = 0.00087.

2: A TCP/IP router has three bidirectional lines and used 247 routes in the last T time. It receives and transmits entire packets of TCP/IP (ie: this is a simple router which performs no partial packet handling). The routes here refer to the variety of destination TCP/IP addresses. 247 different such addresses were used in the last T time so its variety of use was 247 in that respect. The router has six components: the three lines, the routing logic, the routing table upgrade logic and the SNMP polling response logic. The routing table was upgraded 14 times in the last T time, so had variety of use = 14. It was SNMP polled once in this time. The complexities of each of the six components are as follows:

line 1: C = log(1.002) = 0.00087 (variety of use = 2, P = 0.001)
line 2: C = log(1.0021)= 0.00091 (variety of use = 2, P = 0.00105)
line 3: C = log(1.0008)= 0.00035 (variety of use = 2, P = 0.0004)

routing logic: C = log(1 + 247 * 0.00001) = log(1.00247) = 0.0011

routing upgrade logic: C = log(1 + 14 * 0.00004) = log(1.00056) = 0.0002

SNMP response logic: C = log(1 + 1 * 0.0000001) = log(1.0000001) = 0.0

Total router complexity = 0.00343 (sum of the component complexities)

3: The TCP/IP router above has added to it a bidirectional ATM line and software to handle connectionless server and gateway operations. The gateway requires one layer of overlap between TCP/IP and ATM protocols. The ATM logic requires three layers. We therefore add the following five components: the three ATM layers, the line and the gateway layer logic. The probabilities of failure have been multiplied by the variety of use. The values of 1+ P * variety of use for each component are estimated as follows.

ATM cell level: C = log(1.000000002)

ATM call level: C = log(1.00000001)

ATM packet assembly/disassembly level: C = log(1.0000003)

ATM line: C = log(1.000000000001)

gateway layer: C = log(1.0004)

This devices would therefore have most of its probability of failure caused by its gateway layer component. Its other components have minute complexities in comparison.

Discussion of complexity.

The complexity of a device combines the complexity of its parts. Very reliable parts contribute almost nothing to the complexity. Very rarely used but unreliable parts also contribute very little. Commonly used and unreliable parts will dominate the sum, as they should. Moveover, the complexity of a network is the sum of the complexity of its parts. The total complexity of a portion of a network (the sum of the parts in that subnet) may predict the probability of a fault in that portion.

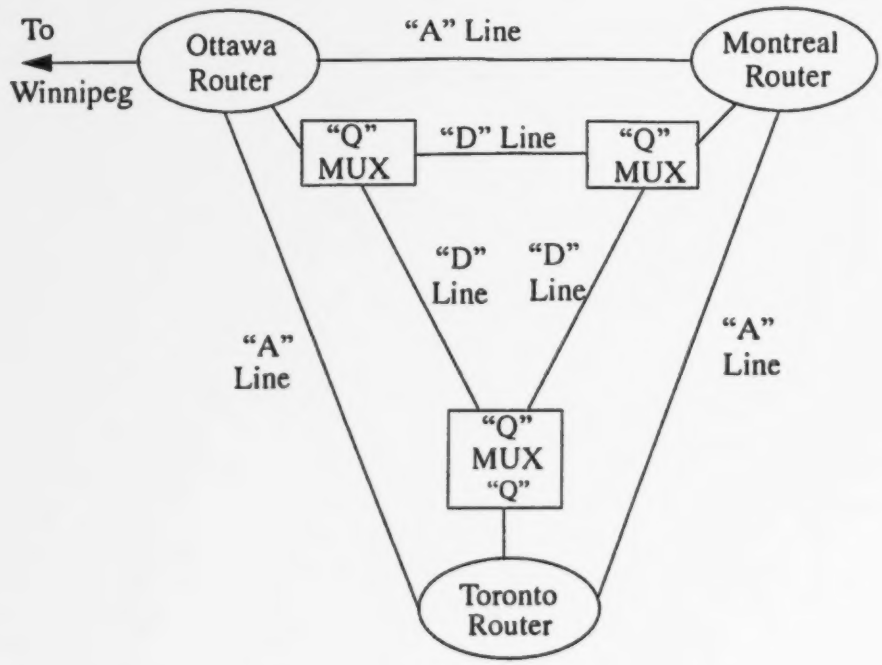

Figure 2

Example Network for Practical Problems